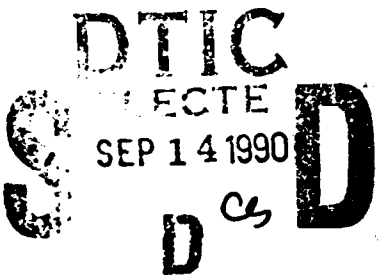
(2)

Technical Report 1362
June 1990

Fractal-Based Image Compression, II



E. W. Jacobs
R. D. Boss
Y. Fisher

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed by the Research Branch, Code 633, Naval Ocean Systems Center, for the Office of Chief of Naval Research, Independent Exploratory Division, Arlington, VA.

Released by
J. C. Hicks, Head
Research Branch

Under authority of
R. H. Moore, Head
ASW Technology
Division

BP

SUMMARY

This technical report comprises a short review of the theory of Iterated Function Systems (IFSs), and a discussion of Recurrent Iterated Function Systems (RIFSs) – an extension of IFS codes which allows for encoding of a far wider set of images. The method and a demonstration of a RIFS-based scheme for automatic compression of images composed of contours is also presented. Finally, a short discussion and a few examples of how these concepts can be used to automatically compress grey scale images is given.

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

The need for data compression is not new. With humble beginnings such as the use of acronyms and abbreviations in spoken and written word, the methods for data compression became more advanced as the need for information grew. The Morse code, developed because of the need for faster telegraphy, was an early example of a data compression technique. Largely because of the growing role of computer technology in todays society, the need for digital data compression (for both storage and communication), is becoming more critical.

In this paper, the specific topic of compression of digital images will be addressed. If one is to consider that just one eight-bit grey scale, 1024 by 1024 pixel image requires a megabyte of storage (or for communication purposes, an appropriate bandwidth and time interval to transfer a megabyte of information), it is easy to understand that, even with the increasing availability of high volume, relatively inexpensive storage medium, there is a great need for developing methods for compression of digital images.

The most fundamental data compression technique is quantization. There are three basic forms of quantization; zero memory quantization, block (or vector or transform) quantization, and sequential quantization. Zero memory quantization is the quantization of one sample at a time. Digitial images are already quantized, thus zero memory quantization can be used to compress images at the cost of resolution (i.e, the number of grey levels per pixel is reduced). Block quantization is a more common technique used to compress digitial images. Instead of individually quantizing each data sample (each pixel), a sequence or block of samples is approximated by the sequence contained in a code book which most closely matches the input sequence. Thus, the number of grey levels need not be reduced, although the total amount of information in the encoded image may be reduced. The optimal block quantizer is the one which has a code book that minimizes the distortion for a given number of quantization levels (i.e., code book entries). When developing a block quantizer, the size of the blocks and the number of quantization levels are fundamental considerations. Similar to zero-memory quantization, where each sample is quantized separately, in block quantization, each block is quantized independently of its neighboring blocks.

The sequential quantization techniques take advantage of the fact that most sources are Markov sources (i.e., consecutive samples are not statistically independent). The most common sequential quantization methods are predictive coders, and the most common of these is called Differential Pulse Code Modulation (DPCM). This method predicts the value of the next sample value (the prediction is based on a weighted combination of previously predicted values) and then quantizes the difference between the predicted value and the actual value.

The measure of the average information in a source is called the entropy (typically measured in bits per sample). Quantization results in distortion of the data. Techniques that result in a loss of information (distortion), are called entropy reduction compression techniques. Redundancy reduction is the name given to a group of tech-

niques that remove redundancy from quantized data without resulting in distortion in the reconstructed data. In the simplest form of redundancy reduction, run length coding, the value of the pixel intensity followed by a fixed length code word containing the number of consecutive pixels with the same intensity is stored. Redundancy reduction techniques that attempt to reduce the average word length (i.e., number of bits in the code for a given sample) to the entropy of the source are called optimum source coding (or entropy coding) techniques. The most common of these is Huffman coding. Methods such as Huffman coding require the use of a table to define the meaning of each code word. (In simple run length coding this is not necessary, because the code word is a fixed number of bits and is defined to be the length of the run).

There are a variety of different methods based on block and sequential quantization (and a host of acronyms to identify them). While the development of variations on the these methods is an active area of research, the fundamental concepts are well established. With the work of Mandelbrot, the concept of representing images (specifically natural images) using non-Euclidian shapes, which he termed *fractals*, was introduced (Mandelbrot, 1977). Mandelbrot describes techniques such as fractional Brownian functions, with which it is possible to easily generate images such as coastlines and mountains which have startling detail and are qualitatively quite similar to real images. The advantage of these techniques is that they utilize compact algorithms for generating fractals, objects of infinite detail, and thereby capture the detail and nature of complex real images without requiring retention of a large amount of information. The deficiency of these techniques, and the problem that is relevant to the image compression issue, is that they are not useful in accurately encoding and subsequently regenerating a specific image.

Recent research in the image compression field has been directed toward taking advantage of the fractal character of images. By using wavelet transforms to perform a multiscale analysis of edges, Zhong and Mallat (1990) have developed a promising new compression algorithm. Another new approach, an approach which evolved directly out of the study of fractal objects and nonlinear chaotic systems, is the work of Barnsley (1988a). The cornerstone of Barnsley's work is the Collage theorem, which is used as a guide in determining the necessary codes (which collectively form an iterated function system or IFS) for regenerating an approximation to a specific image. The IFS is in a sense, similar to a nonlinear dissipative system that possesses a strange attracting set. By using the collage theorem, it is possible to construct the IFS such that its attracting set is in fact the set of points that make up a desired image.

Although the theory of IFSs allows for one to encode the image such that the decoded image will be an exact duplication of the original image, for the purpose of compression it is advantagous to relax the error criteria such that the reconstructed image will be close to, but not exactly the same as the original image. Therefore, the techniques described in this report can be classified as entropy reduction techniques.

It should be pointed out that the decoded IFS is an attractor. That is, one starts with a point (or set points) chosen at random (the points may be inside or outside the final image) and the system is attracted to the set of points which make up the

IFS image. This contrasts with typical fractals (for example the triadic Koch island), which are purely geometric constructs, and not attractors in the sense that the rules of construction do not take a randomly chosen initial state and dictate the evolution of this initial state towards a stable final state. In addition, the images generated by an IFS do not necessarily have fractal character. Labeling the decoded IFS as a fractal is generally correct but it is not a very meaningful description. The term "strange attractor" is usually used in the context of a chaotic dynamic system. Because there are no rules dictating the time evolution of the generation of the reconstructed IFS image, the system is fundamentally different from a dynamic system. Furthermore, the Lyapunov exponent (a fundamental measure of chaotic systems) for a chosen ordering of the IFS iteration is negative, not positive as would be the case for a chaotic process. Consequently, the images resulting from the decoding of an IFS are clearly different than what is typically thought of as a strange attractor or a fractal, and are probably best described as a geometrical attractors.

The following discussion begins with a brief review of the Collage theorem, IFSs, and recurrent iterated function systems (RIFS). In the following section, a solution to the problem of automation of the encoding process in the context of contour images is presented. Results from an implementation of an automated contour image encoder are given. The final section is a brief discussion (with results) of how these ideas can be applied towards the automatic compression of grey scale images.

2. REVIEW OF IFS AND RIFS

2.1 Iterated Function Systems

An iterated function system, W, is defined as

$$W = \bigcup_{i=1}^{m} w_i \,, \tag{1}$$

where each w_i is an affine transform. An affine transform is the result of a set of rotations, skewings, scalings, and translations. The effect of such a transform, w_i, on any point (x_n, y_n) is described by the set of equations

$$\begin{aligned} x_{n+1} &= a_i x_n + b_i y_n + e_i \\ y_{n+1} &= c_i x_n + d_i y_n + f_i \,. \end{aligned} \tag{2}$$

The Collage theorem (Barnsley 1986, 1988a) states that for a set of points (the image), L, and an IFS, $\{w_1, w_2, ...w_N\}$, the Hausdorff distance h (the maximum of all

minimum distances from each point in either image to the other image) between the attractor of the IFS, A, and the set L fulfills the condition

$$h(L, A) \leq \frac{h(L, \bigcup_{n=1}^{N} w_n(L))}{(1-s)} \tag{3}$$

where s is the maximum contractivity factor for the set of transforms w_n. (The contractivity factor is the smallest number such that any pair of transformed points are no farther apart than s times the pre-transform distance. Thus, the maximum contractivity factor is actually a measure of the *weakest* contractivity.) Hence, from the collage theorem, if the transforms are restricted such that they are contractive, there is a guarantee that any iteration on the set of transforms must result in a bounded set of points.

More importantly, the collage thereom states that if the union of the transforms w_n applied to the image L, reproduces the image L to within some error, then there is a known upper bound on the possible error between the attractor of the IFS and the image. By choosing an appropriate IFS, the error between the attractor and the image may be made as small as desired. For a thorough description of the collage theorem and the mathematics upon which it is based see Barnsley (1988). Barnsley and Sloan (1988) have written an article which explains how to perform the encoding and subsequent decoding of simple images. Boss and Jacobs (1989) have demonstrated the application of these concepts for the compression of more general images. A detailed discussion of the encoding and decoding process will not be given here. Such a discussion can be found in the above references.

The examples of images encoded with IFSs (which are shown in the above references) demonstrate some of the capabilities of IFS codes, but IFS codes do have limitations to their usefulness. When encoding an image using IFS codes, one is restricted to covering the original image with copies of itself. The collage theorem guarantees that this can be done, but quite often, it cannot be done with a reasonable number of transforms. For example, the triangle shown in figure 1a is very easy to encode (note the perfect tiling by four transformed triangles shown in figure 1b), but the 'bow tie' shown in figure 2a is very difficult to cover with affine transforms of itself. A solution to this problem will be given in the next section.

2.2 Recurrent Iterated Function Systems

The collage thereom as discussed above applies to an IFS. A more general collage thereom that applies to recurrent iterated function systems (RIFS) has been proven by Barnsley, Elton, and Hardin (1988). The major difference in this technique is that unlike an IFS, where affine transforms of the entire image are used to collage the image, in the encoding process for a RIFS, affine transforms of *parts* of the image are used to cover other parts of the image. It follows that in a RIFS system, during

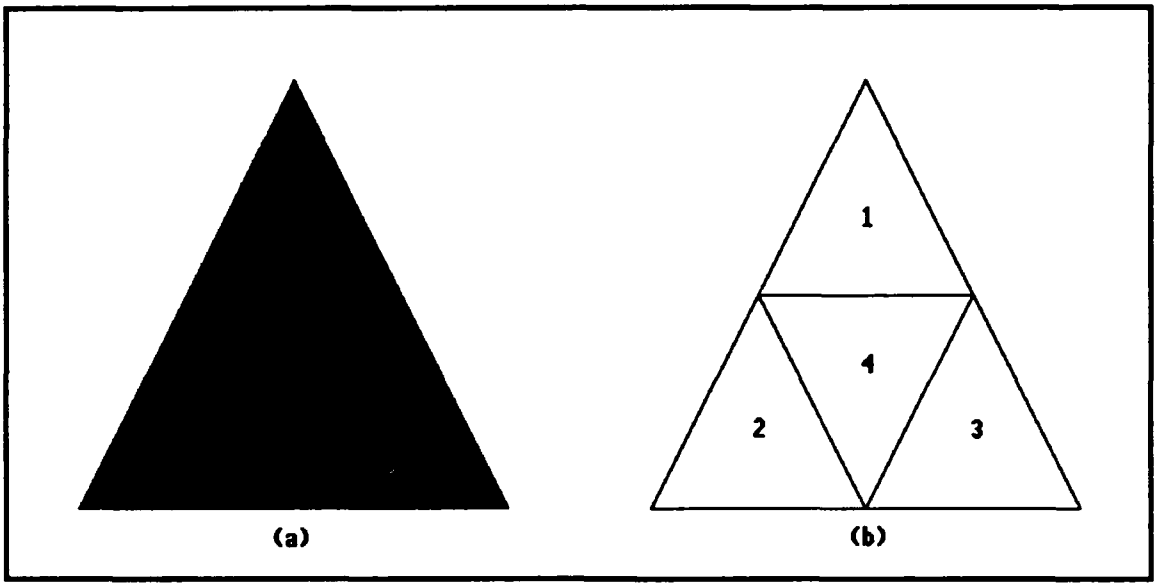


Figure 1: (a) A triangle and (b) the collage of triangles.

the decoding process the allowability of applying a transform is dependent upon which transform was last applied. In fact, after a given transform has been applied (resulting in a point being generated in a specific area of the attractor) only a few transforms may be allowed, with the remaining transforms having zero probability of being applied. This is in contrast to an IFS, where any transform may be applied after any other transform. The added complexity requires that more information per transform be stored, but as will be shown below, the total number of transforms required is often greatly reduced, resulting is a net improvement in the compression.

To illustrate to RIFS technique, the bow tie shown in figure 2a will be encoded. The choice of transforms to be used will not be the most efficient, but will serve to clearly demonstrate the method.

First, consider the tiling of the bow tie illustrated in figure 2b. The first four transforms in the encoded image are those which transform the left-hand side of the bow tie into the four smaller triangles labeled one through four respectively. The second four transforms in the encoded image are those which transform the right-hand side of the bow tie into the four smaller triangles labeled five through eight respectively. These eight transformations are given in table 1.

Referring again to figure 2, it is clear that executing transforms 1 through 4 is allowed only for points residing on the left-hand side of the bow tie. Similarly, executing transforms 5 through 8 is allowed only for points residing on the right-hand side of the bow tie. The transforms that result in points on the left-hand side of the bow tie are transforms 1,2,3, and 8, while the transforms which result in points on the right-hand side of the bow tie are transforms 4 through 7. This information is summarized in the table 2. The transformations given in table 1 combined with the connection rules in table 2 represent the encoded version of the bow tie.

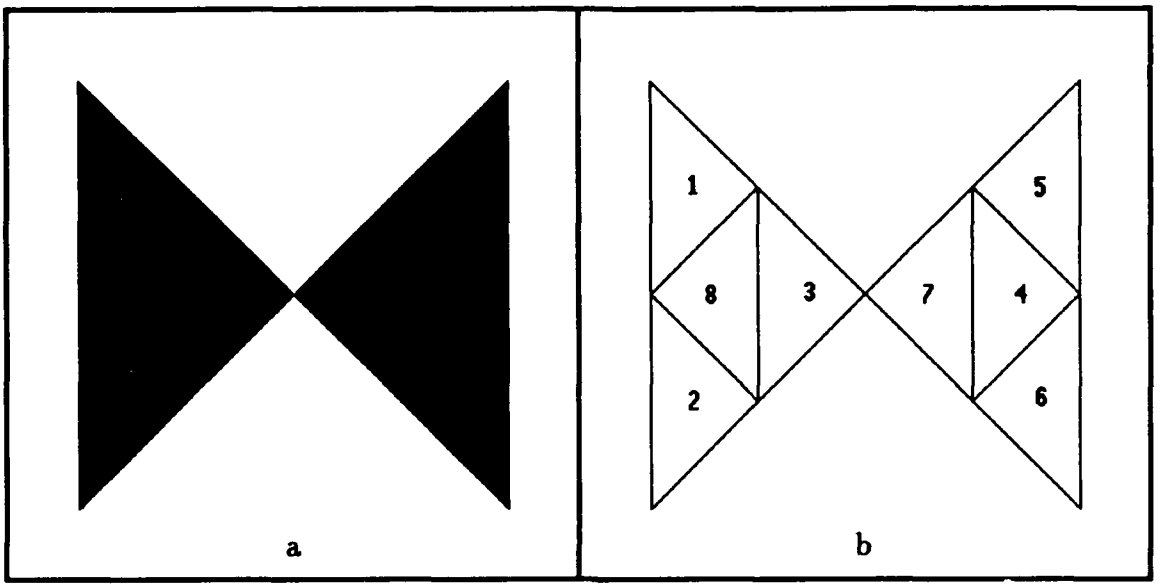


Figure 2: (a) A bow tie image and (b) a collage of parts of image (a).

Table 1: Transforms to make figure 2.

i	a	b	c	d	e	f
1	0.5	0.0	0.0	0.5	-0.5	0.25
2	0.5	0.0	0.0	0.5	-0.5	-0.25
3	0.5	0.0	0.0	0.5	0.0	0.0
4	0.5	0.0	0.0	0.5	1.0	0.0
5	0.5	0.0	0.0	0.5	0.5	0.25
6	0.5	0.0	0.0	0.5	0.5	-0.25
7	0.5	0.0	0.0	0.5	0.0	0.0
8	0.5	0.0	0.0	0.5	-1.0	0.0

The decoding process for this figure is demonstrated in figure 3. Any initial set of points can be used to begin the iteration. In figure 3a, we arbitrarily choose a solid square as the initial image. Furthermore, since no transform has yet been performed, we arbitrarily assume that the square is on the left-hand side of the bow tie. Following the rules dictated by the connection matrix, we apply transforms 1 through 4 to the square. The resulting first iteration of the image is shown in figure 3b. For the second iteration, the connection matrix requires that transforms 5 through 8 be performed on the section of the image resulting from transform 4, and transforms 1 through 4 be performed on the other sections of the image. The second iteration of the image is shown in figure 3c (the grey boxes come from the transforms 5-8, the shading is only to indicate which transform was last applied). Continuing the process, the third iteration of the image is shown in figure 3d. After just three iterations it is clear that the image is converging towards the image of the bow tie.

Table 2: Transformation connection matrix for figure 2.

	FROM LEFT SIDE	FROM RIGHT SIDE
TO LEFT SIDE	1, 2, & 3	8
TO RIGHT SIDE	4	5, 6, & 7

3. AUTOMATED ENCODING APPLIED TO CONTOUR IMAGES

The proceeding sections have described techniques which can result in the storage of data/images with substantial compression, however, the key step of the encoding process must be done interactively, with a person performing the pattern recognition (i.e., choosing the appropriate transformations to encode the image). While this does not preclude the application of these techniques to various problems, it does restrict the variety of problems to which they may be applied (as examples; compression of satellite images prior to transmitting them to earth precludes having a person in the loop, but using an IFS technique to compress a map is a sufficiently finite task, with no temporal restrictions, that having a person/people perform the encoding would not necessarily be restrictive). Nonetheless, for the described fractal compression techniques to be widely applicable it would be necessary to automate the encoding step, that is, to have a computer do the encoding of the image.

The RIFS technique introduced in the previous section represents a major improvement over the IFS technique in that a far wider variety of images can be encoded. The bow tie would be quite difficult to encode using an IFS, but as demonstrated in the in the example, was easily encoded using a RIFS. An improvement of equal importance is that the RIFS technique breaks up the encoding problem into smaller pieces, thus enabling one to address the important problem of automation. In this section, a description of a solution to the automated encoding problem as applied to images made up of contours is given. Barnsley and Jacquin (1988) have demonstrated the use a RIFS to (manually) encode contours representing the edges of clouds. The method described here is an automated implementation of their model. Before addressing the automation problem, an example of a RIFS as applied to a contour image will be described.

In figure 4, the solid outline represents a simple contour which we would like to encode. The numbered points identified on the contour are referred to as nodes, and the segments between each pair of nodes are referred to as ranges. The underlying concept which leads to compression using the techniques discussed in this report is that it is possible to encode an image by finding parts of the image which look similar, but on different size scales. In figure 4, it is evident that the contour between nodes 4 and 8, and the contour between nodes 8 and 9, fit this criteria. Remembering that

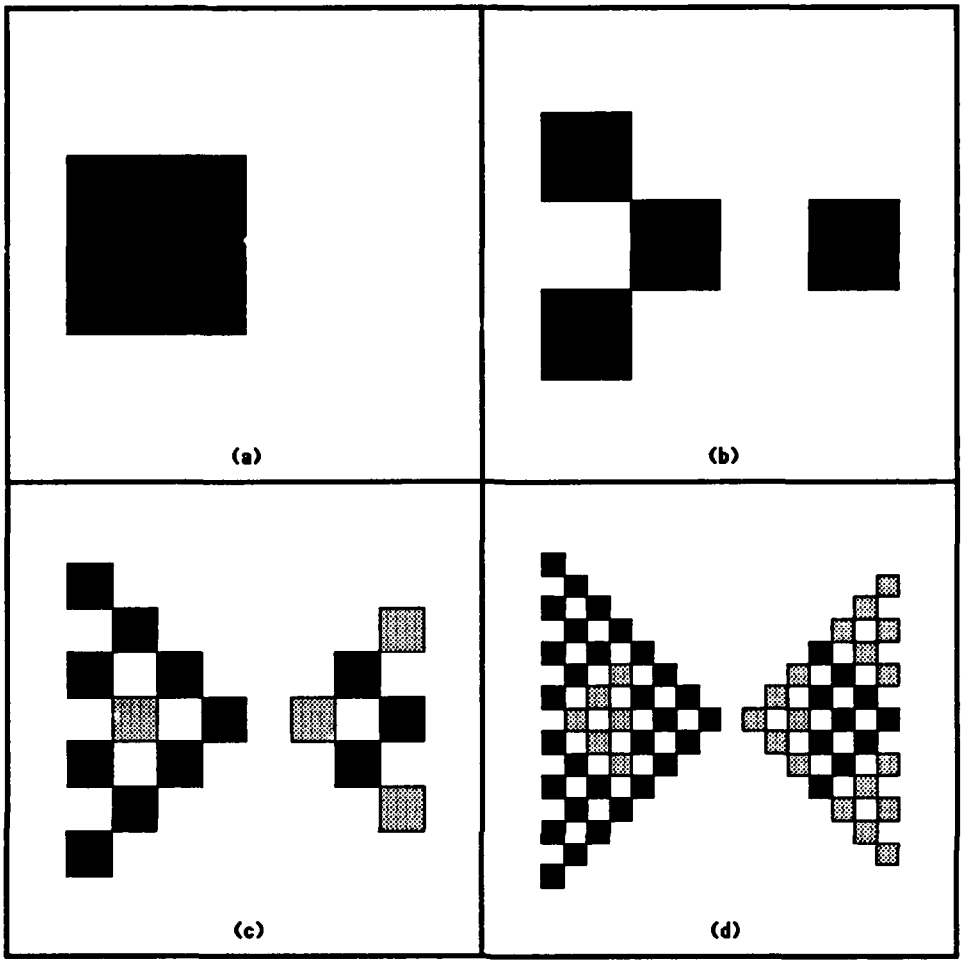


Figure 3: (a) A randomly chosen initial condition - a square, (b) the image after one iteration (using only transforms from the left side of the bow tie, (c) after two iterations (the grey blocks come from the transforms from the right side), and (d) after three iterations.

transformations must be contractive to assure convergence, let us choose to transform the segment between nodes 4 and 8 (the domain nodes) to a contour between nodes 8 and 9 (the range nodes). There are six coefficients in the affine transform. Four of the six coefficients are determined in order to map the domain nodes on to the range nodes. The last two coefficients can be determined by using a least squares minimization of the error between the image and the collage (i.e., the transformation of the domain on to the range). In a similar way, by mapping series of ranges (the domains) onto each of the ranges, the image can be collaged.

Table 3 summarizes the information that composes the encoded image. The first item that needs to be stored is c, the number of contours in the image. The next stored items are n_i, a list of the number of range nodes on each contour. In this example, c equals 1, and n_1 equals 9. The next item that needs to be stored is an ordered list of the n_i (x, y) coordinates of the range nodes for each contour. It is clear from the previous paragraph that it is required that all domain nodes also be range nodes. The next information that is required is an ordered list of the n_{i-1} domain node

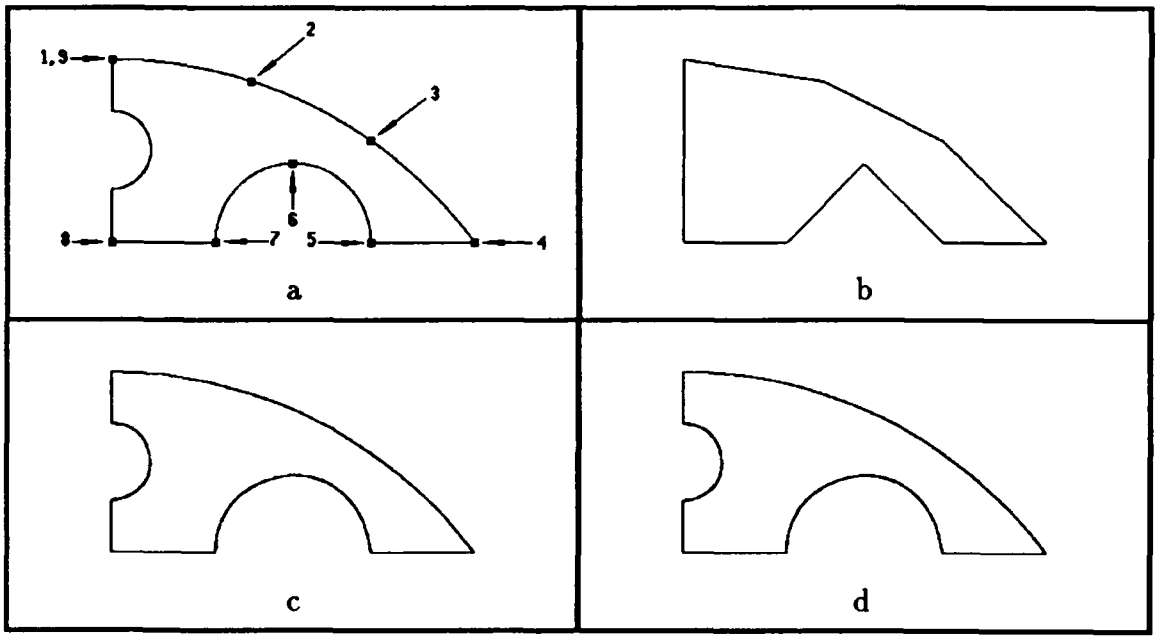


Figure 4: (a) An example contour image with nodes shown and numbered, (b) the zeroth iteration of the RIFS of the image, (c) the first iteration of the RIFS of the image, and (d) the final attractor for the RIFS.

numbers for each range segment. The ordering of the list determines the two range nodes to which the domain nodes are mapped by the transformation. The remainder of the transformation is composed of two additional parameters (determined by least squares minimization) that are needed to completely define the affine transformation. This information represents the encoded image, and from it the original image may be regenerated.

The decoding process is shown in figure 4. As explained in the previous paragraph, the coordinates of the nodes are stored as part of the encoded image. Therefore, the nodes and the lines connecting them, can be used as the starting point of the iteration (this is shown in figure 4b). From table 3, the first part of the iteration is to map the segment between nodes 1 and 4 to the segment between nodes 1 and 2. Next the segment between nodes 1 and 4 is mapped to the segment between nodes 2 and 3. This process is continued for the entire list of transformations. Figure 4c is the resulting image after one complete iteration. Because the transformations used in this example are highly contractive, after just one iteration, the image is very close to its stable state (figure 4d).

Two computer programs have been written which implement the image encoding technique described in the previous paragraphs. The first program has an interactive graphics interface which aids the user in selecting the domain and range nodes necessary to encode the image. The second program, the one of interest, performs the encoding automatically. The problem in creating an automatic encoder is finding a good way to classify sections of the image such that segments which look similar, but are of different scales, can be identified. The approach taken here can be described

Table 3: Encoded data for figure 4. Columns indicated by the symbol'*' are data which need not be stored.

Number of Contours	Number of Nodes
1	9

Range Node Number*	x coordinate	y coordinate
1	100	50
2	234	71
3	349	128
4	450	225
5	350	225
6	274	150
7	200	225
8	100	225
9	100	50

Range Node Numbers*	Domain Node Numbers	b	d
1,2	1,4	-0.0972	0.127
2,3	1,4	-0.0391	0.130
3,4	1,4	0.0037	0.151
4,5	1,4	0.0729	0.000
5,6	1,4	0.3506	0.255
6,7	1,4	-0.2527	0.355
7,8	1,4	0.0729	0.000
8,9	4,8	0.5018	-0.002

as follows. A contour was represented by a list of pixel coordinates. Every nth pixel was checked to determine if the contour formed by connecting the nth pixels (with a straight line) curved continueously in the same direction. As long as the contour continued curving in the same direction, and the total curvature did not excede a predetermined limit, the next nth pixel was checked. When the curvature changed direction, or the predetermined maximum curvature limit was exceded, the segment was terminated at the previous nth pixel. The process was then repeated from that pixel until the end of the contour was reached. In this way, the contours that made up the image were divided up into a series of segments. This process was then repeated for n taking on several different values. Thus, the entire image was segmented several times, each time using a different scale for the segmentation. In the examples to be presented in the following section, n was chosen as 4, 8, 16, and 32. The pixel at which the segmentation was started was staggered in a way such that each segmentation was as independent as possible relative to all the other segmentations. In fact the image was segmented twice with n equal 16, the second time starting eight pixels offset from the first segmentation. The process described here is, in essence, a crude multiscale classification of the image. A more elegant multiscale analysis of the image might be achieved by the use of dyadic wavelet transforms (Zhong and Mallat, 1990). A segmentation based on the maxima of the dyadic wavelet transform on different scales would result is a similar image classification.

After the segmentation of the image was complete, the algorithm used the segments in the biggest segmentation class (32 in the examples below) as ranges to be covered by the (domain) segments in the biggest segmentation class. If the contractivity condition, and both a Hausdorff distance criteria, and a rms error distance criteria were satified, the transformation was saved as part of the encoded image. When a satifactory transformation was identified, the segmentation lists were updated as necessary to ensure that ultimately, the ranges would cover the entire image, and all domains would be composed of consecutive ranges (i.e, all domain nodes were also range nodes). For instance, if a segment was covered (i.e, a range was defined), then any potential nodes in all of the segmentations lists between the end points of the covered range were eliminated. After all possible "32" on "32" coverings had been checked, a check of the smaller segmentation classes was performed in a nested loop. The outside loop incremented the range classes ordered from biggest to smallest, and the inside loop incremented domain classes, ordered from biggest to the current range class. Each time an acceptable tranformation was identified which resulted in an alteration to the segmentation lists, the indicies on the nested loops reverted back to the point necessary to recheck any new segments that may have been created.

To assure that the automated collaging would ultimately cover the entire image, any segment 8 pixels or less in length was covered by the best possible contractive transform, whether or not the Hausdorff and rms error criteria were satisfied. If the entire nested loop was executed, and segments were still left uncovered, all uncovered segments were cut in half, and the nested loop was executed again.

An example of the results of this system are shown in figures 5 and 6. Figure 5a is a copy of a 960 by 428 pixel image of Point Loma (the peninsula in San Diego on which NOSC is located). Figure 5b is a decoded image resulting from a manual encoding of the image in 5a. The encoded image of figure 5b required 4944 bits of storage. The Hausdorff distance (the distance of the "worst" point in the decoded image to the target image or visa versa) for figure 5b is 3.0 pixels. Figure 5c was a decoded image resulting from an automated encoding of figure 5a. The automatically encoded image required 9120 bits of storage, and the Hausdorff distance is 2.236 pixels. The amount of data compression varies depending on how the original image is stored. Storing the image as one bit per pixel would require 410,880 bits. Encoding the bit map using an adaptive Lempel Ziv code requires 19352 bits. An efficient packing of the x,y coordinates used to create the original image requires 12700 bits. Using this (which in a sense is already a highly compressed representation of the image) as a comparison, the data compression of the automatically encoded image was 1.4 to 1.

The encoding of an image entails a mapping of sections of the image onto itself. As a result, an increase in the amount of information contained in the original image can often lead to an increase in the data compression of the encoded image. This is evident in the example shown in figure 6. Figure 6a is a copy of an 800 by 800 pixel image of Baja California. Figure 6b is the decoded image resulting from an automated encoding of figure 6a. The encoded image requires 14160 bits of storage, and the Hausdorff distance is 2.236 pixels. Encoding the target image bit map using an adaptive Lempel Ziv code requires 43552 bits. An efficient packing of the x,y coordinates used to create the original image requires 32660 bits. Using this as a

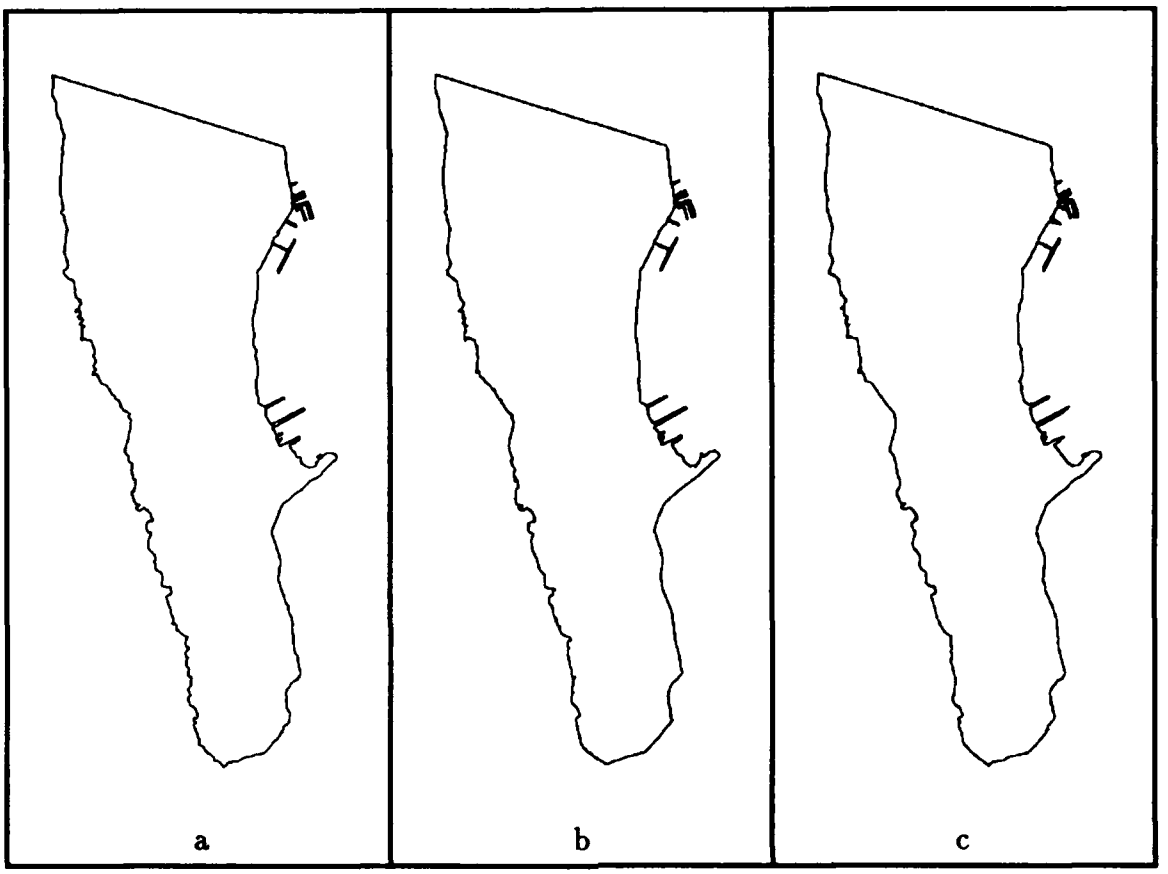


Figure 5: (a) The original image of Point Loma, (b) the manually encoded Point Loma, and (c) the automatically encoded Point Loma.

comparison, the data compression was 2.3 to 1, significantly better than the image of Point Loma.

An important limitation of the contour encoder described here is that the image must be represented by a list of contours, each contour being made up of an ordered list of lit pixels. A program can be written to analyze an image stored as a pixel map and extract a list of contours (and in fact as a front end of the encoder described above, the authors have written code to perform this task), but such a program is quite complicated and in most cases takes longer to run than the encoding process itself. If the encoder described in this section is to find application, it would likely be for applications where the images are already represented in the correct form.

It should be emphasized that the compression ratios quoted above are relative to an extremely efficient representation of the data. Furthermore, there is relatively little information in the contour image to begin with. It is therefore rather impressive that the resulting compression was attained with such small error in the regenerated image.

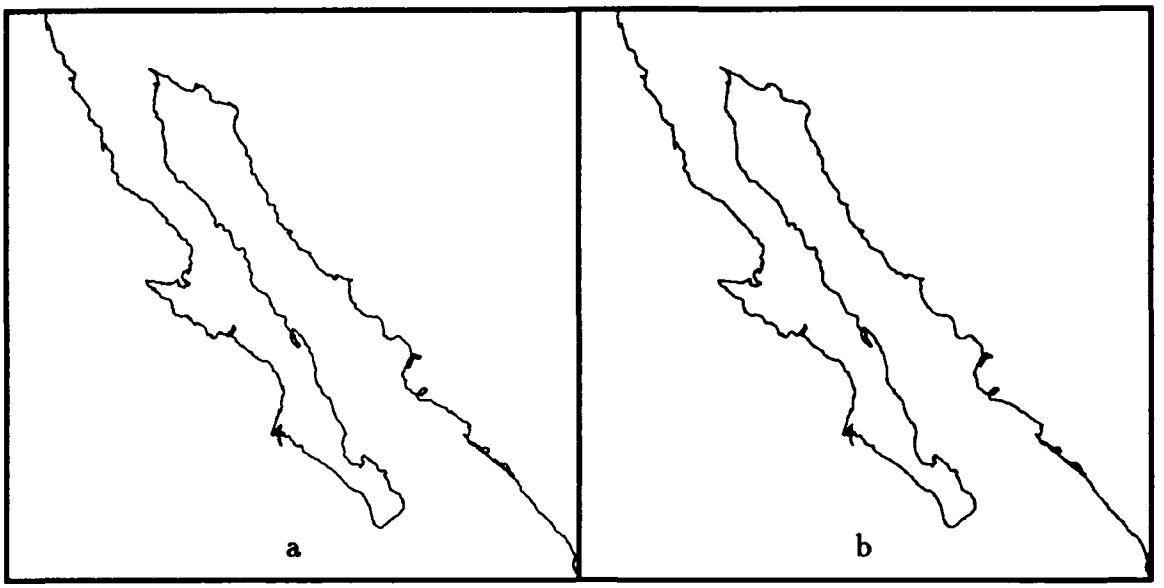


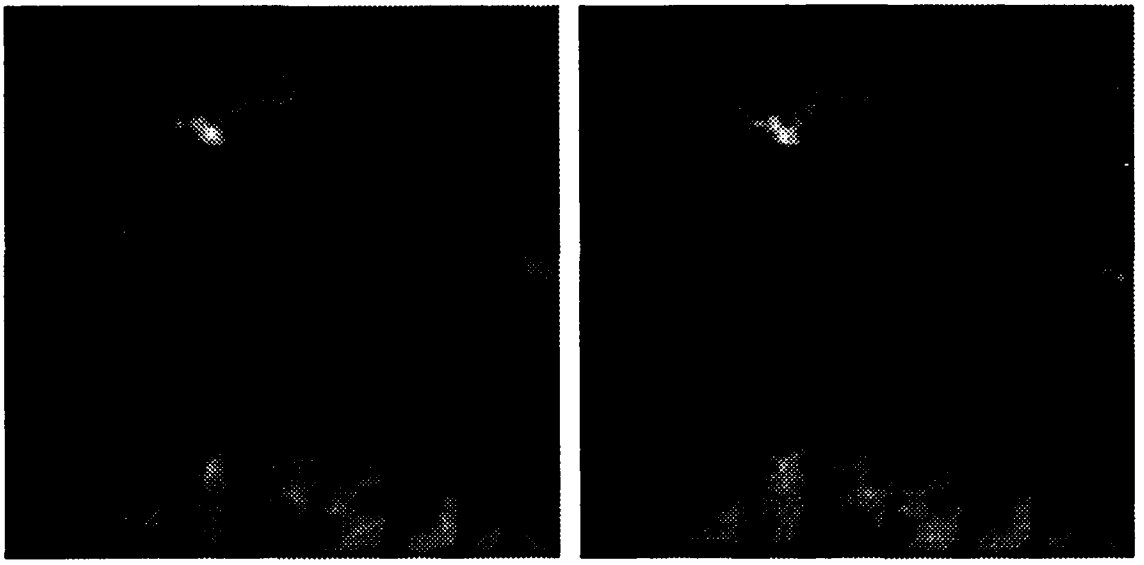
Figure 6: (a) The original Baja California image and (b) the automatically encoded Baja.

4. AUTOMATED ENCODING APPLIED TO GREY SCALE IMAGES

Athough the automated encoding of contour images described in the previous section results in good compression, applications for such a system are relatively limited. The important question to address is that of compression of grey scale images. The applications for an improved method of compressing grey scale images are general and wide spread. Jacquin (1989) has demonstrated the automated encoding of 256 by 256 pixel 6-bit (i.e., 64 levels of grey) grey scale images using a technique based on iterated transforms. The reports of progress by Barnsley and Sloan (Iterated Systems Inc.) using variations of these (and possibly other proprietary) techniques to encode grey scale (and color) images are unclear, but nevertheless very impressive (Science, 1989 and Scientific American, 1990).

Figure 7a is a representation of a section of the Digital Terrain Elevation Data Base (in this case, an area in the fjords of Norway, Long. 5° - 5° 25.6'E, Lat. 61° - 61° 12.8'N). Elevation is represented as different tones of grey, white being the highest elevations, black the lowest (which in this image is ocean). The image is a 256×256 pixel, 8-bit grey scale image. The image in figure 7b is reconstructed from an automatically encoded version of figure 7a. The encoding was accomplished using the same concepts as those detailed in the previous section for the encoding of contour images. The compression of the encoded image was 23:1. The rms error per pixel of the reconstructed image was 7.8, resulting in a signal-to-noise level of 30.3 db.

Images of human faces are commonly used to evaluate image compression techniques because they require a robust compression algorithm. The image in figure 8a is a standard test image. It also is a 256×256 pixel, 8-bit grey scale image. Fig-



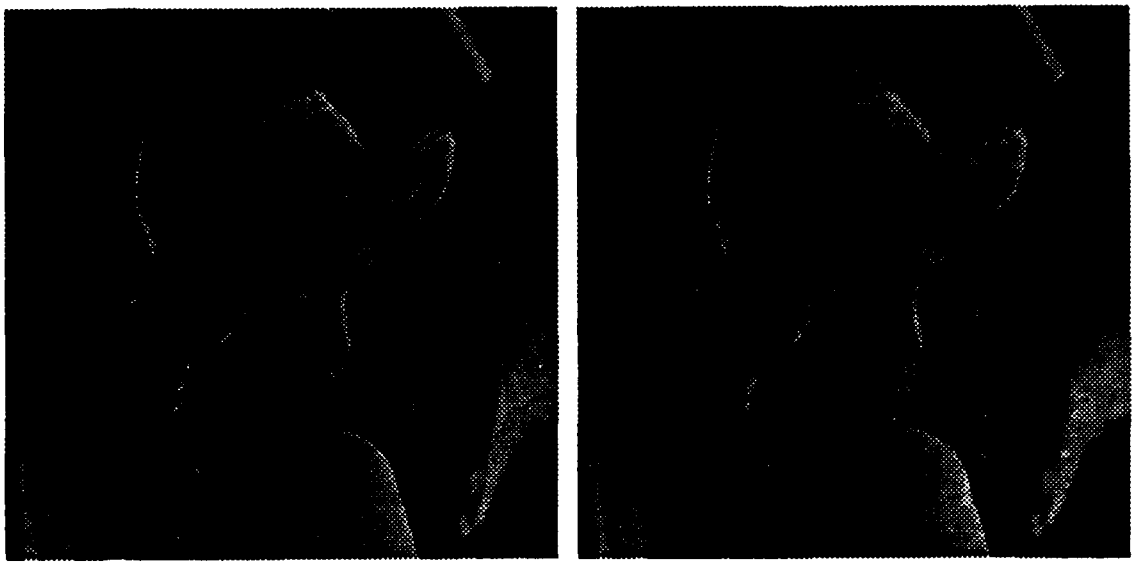
a b

Figure 7: (a) Elevation map of a section of Norway and (b) a compressed version of (a).

ure 8b is reconstructed from an encoded version of figure 8a. The compression of the encoded image was 9.8:1. The rms error per pixel of the reconstructed image was 9.16, resulting in a signal-to-noise of 28.9 db. These results are comparable to more conventional image compression techniques, but are impressive in light of the fact that these represent initial results, and with further work improvement in both the compression and accuracy of the images can be expected.

Another class of images of interest is that of man-made objects. Figure 9a shows a 512×512, 8-bit grey scale image of an aerial photo of El Toro Marine Corps Air Station. The image shown in figure 9b is the reconstructed image after a compression of 10:1. The reconstructed image has an error of 12.3 rms or a signal-to-noise of 26.3 db. The figures 7-9 clearly show the robustness of the method to a wide variety of image classes.

The bridge between automatic encoding of contours, and the method used here to encode grey scale images is not obvious, but is in fact conceptually simple. The contour images dealt with in section 3 of this report are two dimensional objects, i.e., one must specify the x and y coordinate for each pixel in the image that is turned on. The grey scale images discussed in this section are three dimensional objects, i.e., one must specify two spatial coordinates plus the value of the grey scale for each pixel in the image. Therefore, the first step towards encoding grey scale images is generalizing equation 2 such that each transform, w_i, acts upon the three coordinates (x, y, z) according to the equation



a b

Figure 8: (a) A test image of a human face and (b) the compressed version of (a).

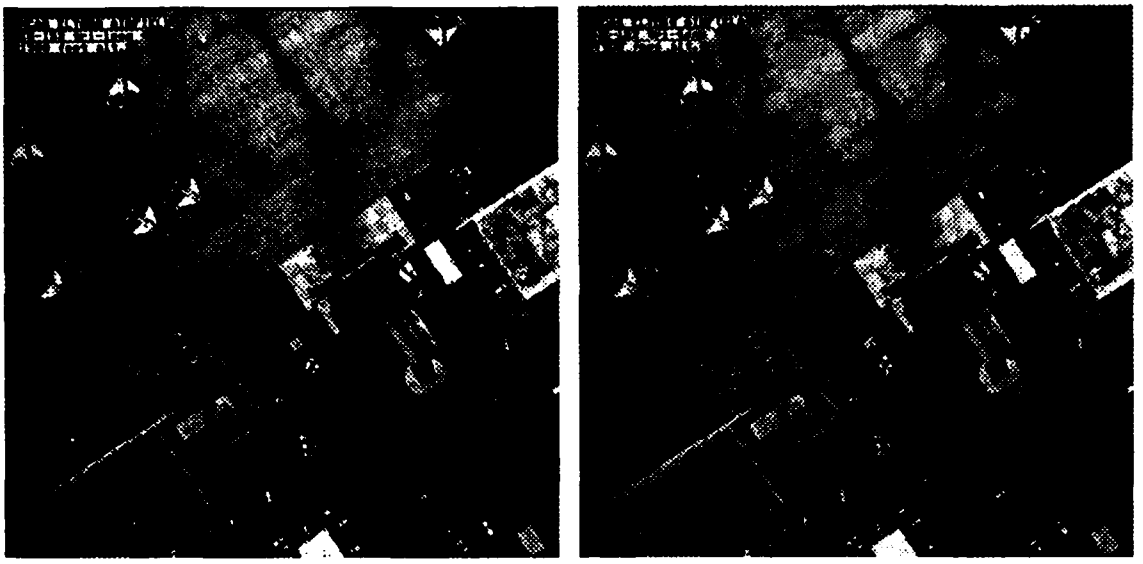
$$x_{n+1} = a_{xx,i}x_n + a_{xy,i}y_n + a_{xz,i}z_n + b_{x,i}$$

$$y_{n+1} = a_{yx,i}x_n + a_{yy,i}y_n + a_{yz,i}z_n + b_{y,i} \qquad (4)$$

$$z_{n+1} = a_{zx,i}x_n + a_{zy,i}y_n + a_{zz,i}z_n + b_{z,i}$$

where z represents the value of the grey scale for each pixel. The collage theorem still holds for systems of dimension greater than two, so, in principle, the problem can still be solved. Unfortunately, the three dimensional affine transform given by equation 4 is an extremely general transformation. The task of finding the best set of general three-dimensional transformations that encode an image would be a monumental computational task, even if a framework for such a computation could be developed. To automatically encode the images shown in figures 7 and 8, restrictions were imposed on the coefficients of the general affine transform, thereby greatly reducing the number of possible transformations. These restrictions served two primary purposes. First, they facilitate faster identification of a good set of transforms to encode the image (the images in figures 7 and 8 were encoded in less that three minutes on a Convex computer), and second, they allow for efficient storage of the encoded information (i.e., a more general transformation will require more information to be stored to define it). A detailed description of the model and implementation of the encoding algorithm for grey scale images will be published separately (Fisher, Jacobs, Boss, 1990).

Before concluding, it should be noted that the problem of encoding of color images (which are in fact the combination of a grey scale image for each of the three primary



a b

Figure 9: (a) An image of El Toro MCAS and (b) the compressed version of (a).

colors) is essentially solved by the encoding of grey scale images. In fact, one can expect that increased compression will be achievable when encoding color images because of the similarity between the images which make up each color plane. One may take advantage of this similarity to encode the image more efficiently.

The fractal-based compression techniques discussed here are new and powerful, and their potential applications are numerous. The progress shown here and by others working on these techniques provides motivation for continued work in this area.

5. REFERENCES

Barnsley, M.F., Ervin, V., Hardin D., and Lancaster J., 1986, *Proc. Natl. Acad. Sci. 83*, 1975.

Barnsley, M.F., 1988, *Fractals Everywhere*, Academic Press, Inc., San Diego.

Barnsley, M.F., and Sloan, A.D., 1988, *Byte 13*, 215.

Barnsley, M.F., Elton, J.H., and Hardin D.P., 1989, *Constr. Approx. 5*, 3.

Barnsley, M.F., Jacquin, A.E., 1988, *SPIE 1001, Visual Comm. and Image Processing*, 122.

Boss, R.D., and E.W. Jacobs, 1989, *NOSC TR-1315.*

Fisher, Y., Jacobs, E.W., Boss, R.D., *to be published in IEEE Trans. on Acoustics, Speech and Signal Processing.*

Jacquin, A.E., 1989, *Ph. D thesis*, Department of Mathematics, Georgia Institute of Technology.

Mandelbrot, B.B., 1977, *Fractals, Form, Chance, and Dimension*, W.H. Freeman and Company, San Francisco.

Science, 1989, *243*, 1288.

Scientific American, March 1990, 77.

Zhong, S. and Mallat, S., *preprint*, 1990, see also Mallat, S.G., *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1989, *37*(12), 2091-2110.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1990	Interim

4. TITLE AND SUBTITLE

FRACTAL-BASED IMAGE COMPRESSION, II

5. FUNDING NUMBERS

PE: 0602936N
PN: RV36
TN: ZE88
WU: DNIC000037

6. AUTHOR(S)

E. W. Jacobs, R. D. Boss, Y. Fisher

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Ocean Systems Center
San Diego, CA 92152-5000
Code 633

8. PERFORMING ORGANIZATION REPORT NUMBER

NOSC TR 1362

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Chief of Naval Research
Arlington, VA 22217-5000
OCNR-20T

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A short review of the theory of IFSs and a discussion of Recurrent Iterated Function Systems (RIFS) which is an extension of IFS codes. How use of RIFSs allows for encoding of a far wider set of images. Automatic compression of image composed of contours using an RIFS scheme is also presented. A short discussion of how these concepts can be used to automaticallly compress grey scale images is given.

14. SUBJECT TERMS

Iterated Function Systems (IFS)
Recurrent Iterated Function Systems (RIFS)

15. NUMBER OF PAGES

24

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT